

(43) Date of A Publication 29.12.1999

(21) Application No 9905578.1

(22) Date of Filing 22.03.1999

(30) Priority Data

(31) 09056547 (32) 07.04.1998 (33) US

(71) Applicant(s)

Hewlett-Packard Company  
(Incorporated in USA - Delaware)  
3000 Hanover Street, Palo Alto,  
California 94303-0890, United States of America

(72) Inventor(s)

Binnur Al-Kazily

(74) Agent and/or Address for Service

Carpmaels & Ransford  
43 Bloomsbury Square, LONDON, WC1A 2RA,  
United Kingdom

(51) INT CL<sup>6</sup>

G06F 11/30

(52) UK CL (Edition Q)

G4A AFMD

(56) Documents Cited

EP 0822498 A1 US 5872931 A US 5655081 A

(58) Field of Search

UK CL (Edition Q) G4A AFMD AFMX  
INT CL<sup>6</sup> G06F 11/30 11/32 11/34, H04L 12/26  
Online: WPI, EPODOC, COMPUTER

(54) Abstract Title

**Diagnostic reporting system for networked peripheral components uses software agents**

(57) Diagnostic reporting system (10, Fig. 1) to monitor the functionality of peripheral components (24, 26, 28, Fig. 1) such as printers, multi-function peripherals, servers, workstations etc. within a computer network environment (12, Fig. 1). The system comprises a number of subsystems 14, 16, 18, e.g. a print server subsystem, and each subsystem contains a number of scout "Bots" 20, which are self-contained, independent, software entities or agents, each being configured with object-oriented design techniques to collect and analyse information and report problems relating to a particular subsystem component, and to report solutions to a detected problem. Each subsystem has an expert "Bot" subsystem controller 40, which keeps track of the associated Scout "Bots" 22 and uses an expert system 42 to send subsystem diagnostics information to a master controller computer 25, in turn having an expert "Bot" diagnostic reporting system controller 30 which uses an expert system 32 to provide system diagnostic reports.

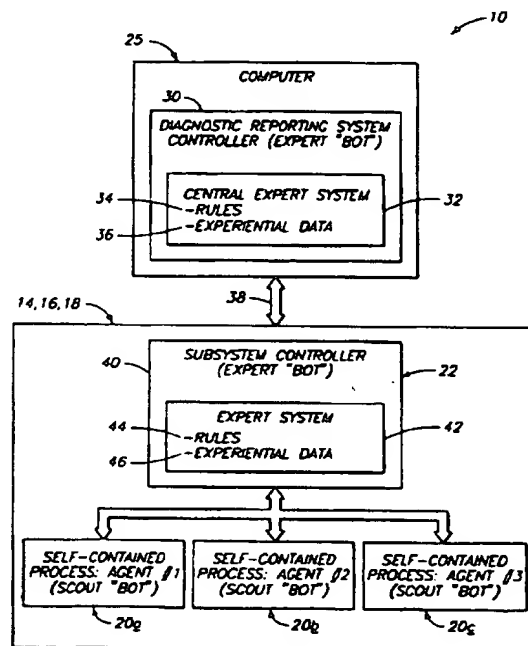
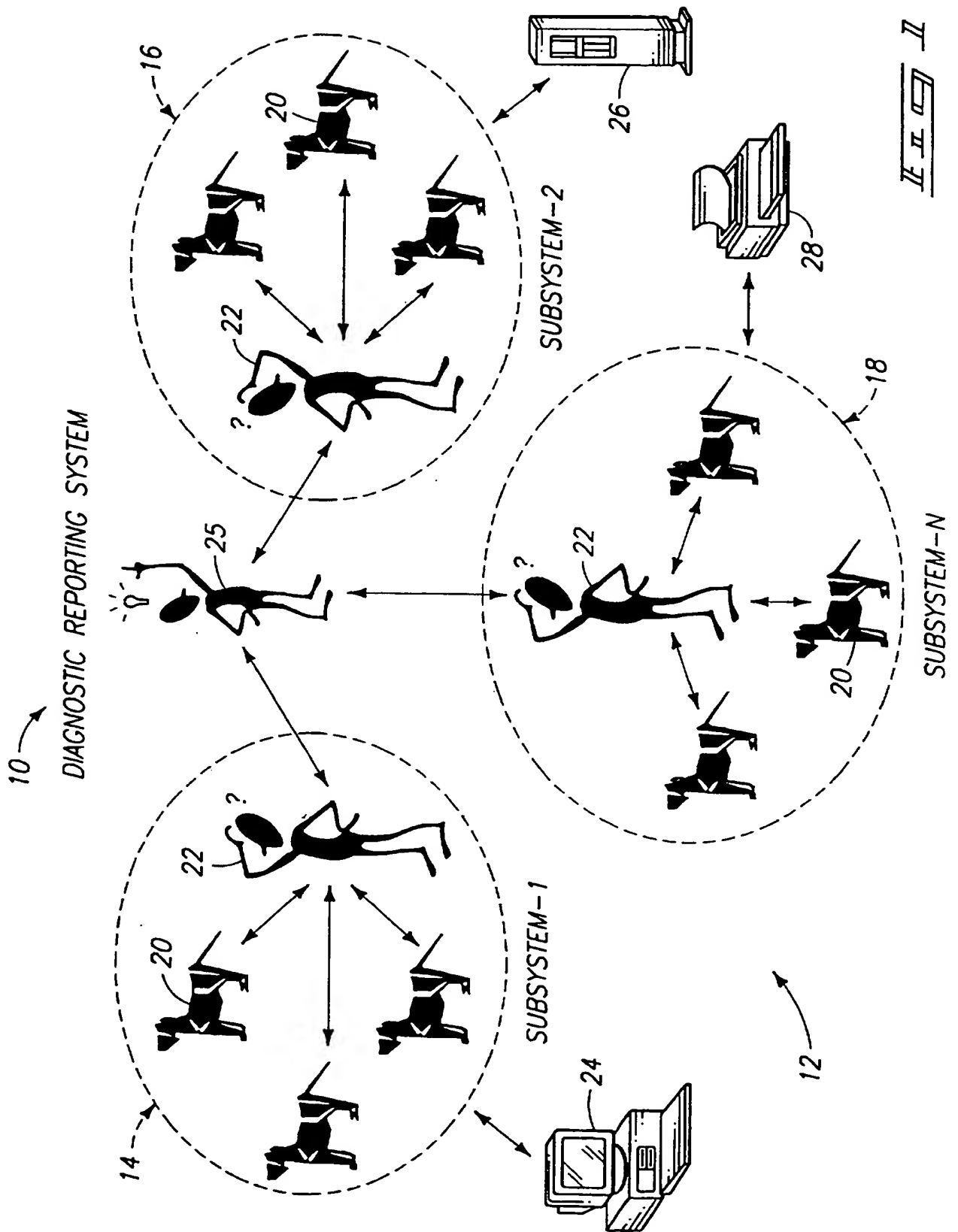
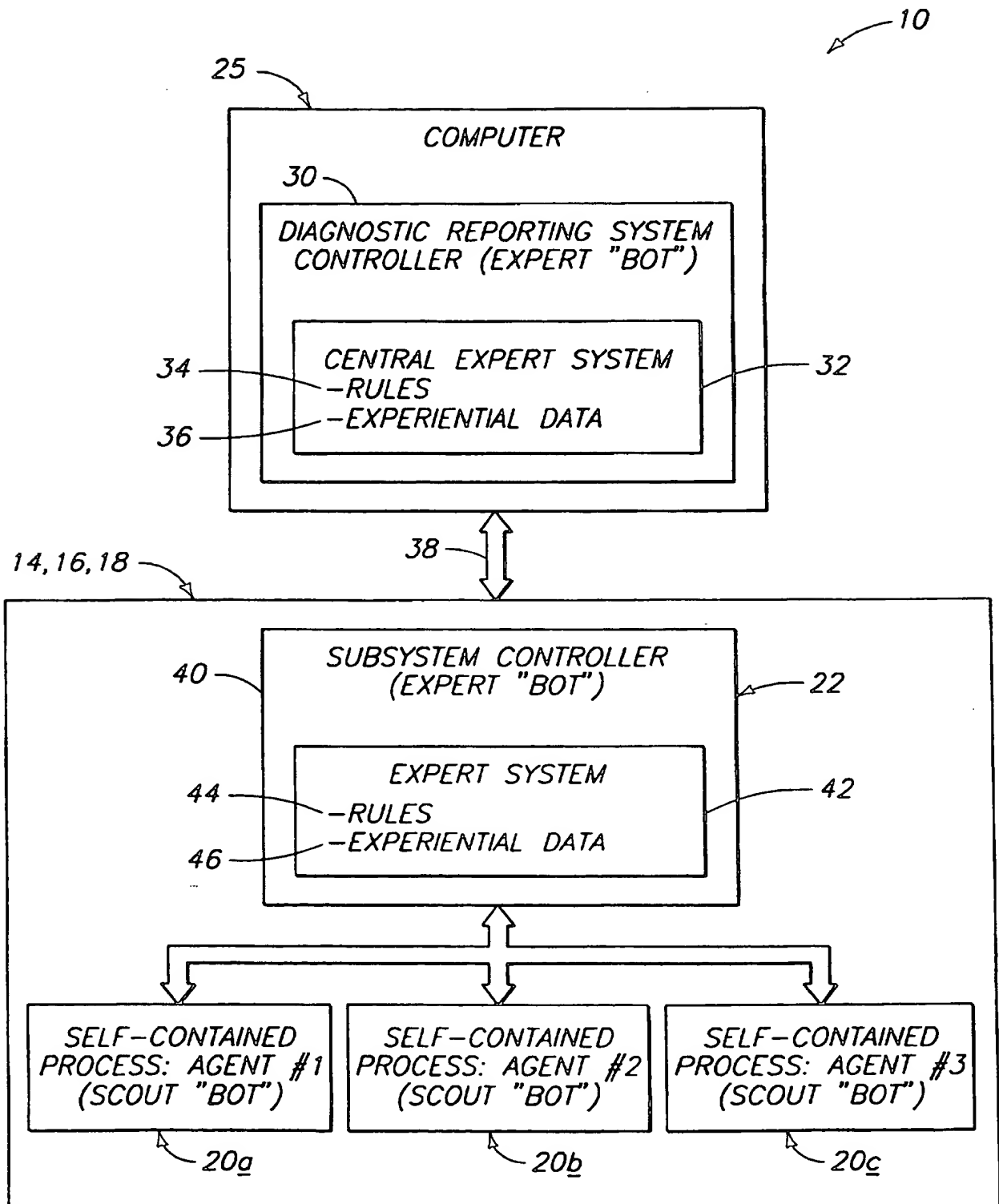


FIG. 1





## DESIGNING A DIAGNOSTIC REPORTING SYSTEM USING SCOUT "BOTS"

### FIELD OF THE INVENTION

5           The present invention is directed to peripheral devices, and more particularly to a diagnostic reporting system for peripheral devices such as printers and multi-functional peripherals used with enterprise-wide network management solutions.

### BACKGROUND OF THE INVENTION

10           The ability to monitor the performance of components within a computer system such as an enterprise-wide network containing peripheral devices such as a printer has been widely proposed and is reasonably well understood in the art. In general, peripheral devices such as printing systems exist as printing clients and servers. A client is a workstation or personal  
15   computer placed in a client/server environment. A printing client is a printer placed in a client/server environment. A server is generally understood to be a computer that is placed in a network shared by multiple users, including file servers and print servers. For the case of a print server, a network computer is configured to control one or more printers. However, present printing clients  
20   and servers are not configured to be self-aware. In other words, printing clients and servers do not know how to identify and report problems to users or system administrators. The problem is even greater in network printing environments, where the printing path extends from a client machine, to possibly a print server(s), and finally to individual printing devices.

25           One technique for diagnosing and reporting an operating problem for a printer involves the provision of status lights on a control panel of a printer that visually alert a user to a printer's status. For example, a Hewlett-Packard 4L printer contains four status lights on a front control panel: one (READY) configured to alert a user when the printer is ready to print a jog; another  
30   (DATA) configured to alert a user when the printer is receiving or processing

5 data; yet another (PAPER) configured to alert a user when a paper cassette is  
empty or missing, or when there is a paper jam; and another (ERROR) configured  
to alert a user when a top door is open or a toner cartridge is improperly  
installed. Additionally, Hewlett-Packard Explorer software enables a user to  
send commands to a printer from a personal computer, and to view printer  
10 status on the personal computer screen.

Another technique involves the use of Hewlett-Packard Openview  
network management software. Openview is an enterprise-wide network  
management solution that provides limited node management solutions,  
including the ability to diagnose limited printer problems remotely from the  
15 printer. More particularly, a control menu application is enabled by a network  
computer that creates a window display, enabling display and selection of  
associated network printers contained within the operating environment. A  
device property page presents basic information about the selected printer,  
including status information. A second property page is dedicated for  
20 diagnostics which provides configuration information as well as statistical data.  
For example, a control panel field within a device property page can show a  
message indicating "PC TRAY EMPTY". However, such Openview  
implementations are realized as serial implementations that require a user to  
manually "poll" a device by accessing each dedicated device property page and  
25 associated control panel field(s).

The emergence of new technologies, and the introduction of new  
ways of doing things to existing packages, has increased the need for diagnostic  
tools. Concurrently, increases in the complexity of computer operating systems  
and environments has lead to an increase in the number of components within  
30 an environment requiring diagnosis and reporting. In the past, these needs have  
been met individually by a systems administrator or user by individually  
monitoring each component of an environment, then referring identified  
problems to customer help desks. However, as technology gets more  
complicated, there exist increasing needs to develop new diagnostic reporting

- 5 systems that continue to deliver improvement gains in diagnosing ever-more complicated computer environments.

### SUMMARY OF THE INVENTION

According to one aspect of the invention, an apparatus is disclosed for independently and/or simultaneously monitoring functionality of peripheral  
10 components within a computer network environment. The apparatus includes a computer and a peripheral component signal coupled with the computer. Additionally, the apparatus includes a plurality of agents. At least one agent is associated with the peripheral component and is operative to collect information needed to diagnose functionality of the peripheral component.

15 According to another aspect of the invention, a method is disclosed for diagnosing and reporting peripheral functionality for peripheral components contained within a network environment. The method includes the steps of: providing at least one autonomous agent associated with a peripheral component and operative to monitor peripheral device functionality and capable  
20 of reporting such monitored functionality to a user accessing the network environment; concurrently monitoring functionality of each peripheral component with at least one associated agent; diagnosing peripheral functionality by evaluating the monitored peripheral functionality; and alerting a user to the diagnosed peripheral functionality when a condition warranting the alerting of a  
25 user is determined, for each peripheral device contained within the network environment.

Objects, features and advantages of the invention are to provide a diagnostic reporting system capable of diagnosing subsystem problems in parallel using a self-contained set of processes to perform basic tasks for  
30 information collection, processing and reporting, in a manner that is relatively simple to implement with existing network management software, is relatively cost effective, and requires relatively few additional hardware and software components, and in a manner that does not require a user to individually poll

- 5 subsystem components within an environment to determine system problems that require diagnosis and/or reporting.

### DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified perspective view of one enterprise-wide network associated with a diagnostic reporting system of this invention and illustrating "Expert Agents" that control diagnostics in each subsystem and self-contained "Scout Agents" that troubleshoot given subsystem problems.

Fig. 2 is a functional block diagram of a presently preferred embodiment of the invention illustrating a diagnostic reporting system of this invention for collecting, processing and reporting information on subsystem components within an enterprise-wide network.

### DETAILED DESCRIPTION OF THE INVENTION

This disclosure of the invention is submitted in furtherance of the constitutional purposes of the U.S. Patent Laws "to promote the progress of science and useful arts". U.S. Constitution, Article 1, Section 8.

20 A preferred embodiment of the invention is illustrated in the accompanying drawings particularly showing a diagnostic reporting system for collecting, processing and reporting problems with subsystems of enterprise-wide networks generally designated with the numeral 10 in Figure 1. System 10 generally is implemented on an enterprise-wide network 12 as shown in Figure 1, and more particularly in one implementation as a diagnostic reporting system for printing systems. According to Figure 1, diagnostic reporting system 10 comprises a client installation subsystem 14, a print server subsystem 16, and a print device subsystem 18.

Each subsystem 14, 16 and 18 contains a self-contained set of processes that are configured to perform basic tasks for information collection, processing and reporting. More particularly, a plurality of self-contained software entities, or agents, 20 are configured with object-oriented software design techniques to handle results in parallel with other software entities. Such

5 software entities collect, process and output diagnostic results, as well as make appropriate decisions based on results, with such results being collected by one or more local expert systems 22. The expert systems 22 collect the results from the software entities for each subsystem 14, 16 and 18 and produce decision-based results that make sense to a user. A central expert system 25, 10 configured as a central computer, then collects the decision-based results from each subsystem 14, 16 and 18 and produces further decision-based results based upon the decision-based results and data received from each subsystem 14, 16 and 18.

### **Definition of Terms**

15 **Agent** A program that gathers information or performs some other service without immediate presence of a human and on some regular schedule. For purposes of this disclosure, a software agent is also referred to as a "bot" which is shorthand for a software robot.

20 **Autonomous** Undertaken or carried on without outside control: self-contained; existing or capable of existing independently.

**Bot** Shorthand for "robot" and referring to a program that operates as an agent for someone, often as a searcher of information or monitor of events. The meaning is essentially 25 the same as "agent", but perhaps with the connotation of being somewhat more autonomous or unrelenting.

**Expert System** An expert system is a computer program that simulates the judgement and behavior of a human or an organization that has expert knowledge and experience in a particular field. 30 Typically, such a system contains a knowledge base having accumulated experience and a set of rules for applying the



5 knowledge base to each particular situation that is described to the program.

Intelligent Agent Intelligent agents form a new class of software which can act on a user's behalf. This includes finding, filtering information, and personalizing information for the user.

10 Robot Derived from the Czech word for work, a machine built to resemble a human in behavior, intelligence and sometimes also in appearance. Successful robots tend to be limited to particular functions, like automatic pilots for routine flights of an airplane or machines on an assembly line replacing  
15 operators performing repetitive tasks.

Although this invention is being disclosed as a diagnostic reporting system for printing services, it is understood that the underlying technology can be applied to any computer literate device. For example, the invention can be implemented on multi-function peripherals (MFP's), facsimile machines,  
20 scanners, copiers, memory devices, servers, workstations, or network computers.

According to this invention, the Scout "Bots" approach comprises using a self-contained set of processes that are implemented to perform basic tasks for information collection, processing and reporting. In order to implement  
25 such tasks, a set of steps are necessary in order to design a diagnostic reporting systems having the Scout "Bots" implementation approach. More particularly, the first step taken in designing a diagnostic reporting system of this invention requires the identification of a subset of diagnostic reporting systems as shown in Figure 1. For example, where the diagnostic reporting system is a printing  
30 system diagnostic reporting system, the subsystems in one implementation comprise: a) a client installation subsystem 14 - configured to diagnose client installation components such as on a computer 24; b) a print server subsystem

- 5    16 - configured to diagnose the client communications with a print server 26;  
and c) a print device subsystem 18 - configured to diagnose a print device 28  
status.

10    The client installation subsystem 14 can be configured to scout  
client installation, network package installation, client communication with the  
network, etc. Hence, a self-contained operating program, or Scout "Bot" would  
be provided for checking functionality of each specific subsystem component  
such as client installation, network package installation, and client  
communication with the network.

15    The print server subsystem 16 can be configured to scout the print  
server setup, spooler diagnostics, etc. Hence, a self-contained operating  
program, or Scout "Bot" would be provided for checking functionality of each  
specific subsystem component such as printer server setup and spooler  
diagnostics. A plurality of Scout "Bots" can operate in parallel or in series.

20    The print device subsystem 18 can be configured to scout the  
paper handling paths, device configuration, disk issues, memory problems, etc.  
Hence, a plurality of self-contained operating programs, or Scout "Bots", would  
be provided, one provided for checking functionality of each specific subsystem  
component such as details describing paper handling paths, device  
configuration, disk issues and memory problems.

25    Once the above subset of diagnostic reporting systems, or  
subsystems, have been defined, the next step in the process involves identifying  
a set of Scout "Bots" dedicated for diagnosing each subsystem problem(s). For  
example, a print device subsystem could be provided with the following Scout  
"Bots", each configured to check functionality of a specific subsystem  
30    component: a) one or more device configuration Scout "Bots" - each operative  
to scout problems relating to device configuration issues; b) one or more paper  
handling Scout "Bots" - each operative to scout specific problems relating to  
paper handling issues, such as paper types, out of paper, paper trays, etc; and  
c) one or more disk handling Scout "Bots" - each operative to scout problems  
35    relating to operation/configuration of a disk within the system.

5           Figure 1 illustrates the implementation of the above components. As shown, diagnostic reporting system 10 is depicted as a three-tier, or level, system formed by a first-tier Expert "Bot" 25, a plurality of second-tier subsystem Expert "Bots" 22, and a plurality of third-tier Scout "Bots" 20 provided within each subsystem 14, 16 and 18.

10           The third-tier comprising Scout "Bots" 20 is configured to troubleshoot given, or associated, subsystem problems that might occur with associated subsystem features. For example, one subsystem might comprise disk issues such as problems that relate to disk storage space, problems with physical damage to a disk resulting from damage, etc. As described further  
15 below, each Scout "Bot" is formed from a self-contained software entity that is configured to effectively collect, analyze and report problems, such as disk problems. Additionally, a Scout "Bot" can also be configured to report solutions to a detected problem.

          The second-tier comprising Expert "Bots" 22 is configured to  
20 operate as a subsystem controller. More particularly, such subsystem controller is operative to keep track of the associated, registered Scout "Bots" 22. Furthermore, such subsystem controller is operative to collect, analyze and report the results received from the third-tier Scout "Bots" 20. Therefore, such subsystem controller is operative to provide possible solutions to problems that  
25 the subsystem might have, thereby providing a diagnostic reporting subsystem.

          The first-tier comprising Expert "Bot" 25 is configured to operate as a master controller of the diagnostic reporting system. Such master controller is operative to keep track of the registered subsystems in the diagnostic reporting system 10, and analyze the results reported by each  
30 subsystem 14, 16 and 18. Therefore, such master controller is operative to provide possible solutions to problems that the system might have, thereby providing a diagnostic reporting system.

          Details of two such Scout "Bots" are disclosed at the end of the Detailed Description as "pseudocode" for a "paper level monitoring" Scout "Bot"  
35 and a "paper jam monitoring" Scout "Bot".

5 Scout "Bots" 20 are capable of being run on a client system as separate processes that are not tied directly to any other software, yet provide diagnostic services to a spooler or any other system components. Inter-process communications can be used between the Scout "Bots" 20 and the Expert "Bots" 22 and 25. Scout "Bots" 20 can be used to incrementally extend the capabilities of diagnostic reporting system 10 or subsystems 14, 16 and 18.

As shown in Figure 2, computer 25 forms a central expert system implemented as a diagnostic reporting system controller 30. Controller 30 comprises a central controller of computer 25, including a processor and memory. A central expert system 32 is implemented on controller 30 comprising an inference engine and a knowledge base in the form of software containing a set of rules and data that interacts with a user and processes the results from a set of rules 34 and experiential data 36 provided in the knowledge base. One such expert system comprises a neural network. Another such expert system comprises a fuzzy logic machine, or a neuro-fuzzy logic machine.

One neural network implementation comprises a set of elements that start out connected in a random pattern, and, based upon operational feedback, are reconfigured into a pattern required to generate a set of required results based upon experience. For example, one suitable artificial neural network comprises a single-layer feedback network such as a discrete-time HOPFIELD network.

One fuzzy logic implementation comprises a fuzzy expert system having a fuzzy inference engine containing membership functions based upon a matter of degree, rather than on yes/no characterizations. Such a system contains fuzzy sets, a fuzzy set of conclusions, and a set of rules. However, any of a number of fuzzy logic and neuro-fuzzy logic implementations can be utilized within an expert system of this invention.

As shown in Figure 2, computer 25 communicates with each of subsystems 14, 16 and 18 via one or more communications links 38 in order that information is transmitted between central expert system 32 and each

5 subsystem expert system 42. Each subsystem 14, 16 and 18 is configured with a subsystem controller 40 comprising an Expert "Bot" and a plurality of task-specific self-contained process modules, or Scout "Bots", 20 (a-c). Each subsystem controller 40 includes a dedicated expert system 42, similar to central expert system 32. Each subsystem expert system 42 is assigned rules  
10 44 and experiential data 46 to implement problem solving and decision-making for each subsystem, with decisions being relayed to central computer 25 for processing via central expert system 32. Central expert system 32 then undertakes to alert a user to specific local diagnostic problems that have been detected with each subsystem, in a parallel implementation.

15 Each diagnostic reporting system controller 40 implements an expert "Bot" in the form of a subsystem expert system 42 having a specific set of related rules 44 and associated experiential data 46. One technique for acquiring experiential data 46, with a neural network implementation, is to train the neural network with real-world experiential information and monitor the  
20 resulting outcome. For example, a plurality of detected conditions present on a printer might warrant notifying a user that a service call needs to be performed by a service technician. One condition requiring notification might be triggered when a "toner is low" condition is met, in combination with a "Paper is low" condition, or a "paper is jammed" condition. A neural network can be  
25 configured to determine when a user needs to be notified via computer 25 that a certain condition exists requiring one of several actions, such as "an immediate service call needs to be performed" or "a service call needs to be performed - sooner or later".

#### Diagnostic Reporting System Roles

30 Expert systems 42 each form an Expert "Bot" for one of subsystems 14, 16 and 18 that functions as a subsystem controller via each controller 40. These subsystem controllers 40 each collect results from a dedicated set of associated Scout "Bots" 20 (a-c). Such results are then processed and the results are prioritized and reported to a user, when

5 warranted, along with possible solutions. It is understood that Expert systems  
32 and 42 each comprise an artificial intelligence computer application.

A knowledge engineer is used to design each expert system 32  
and 42. It is understood that a knowledge engineer is an individual, or  
individuals, who studies how human experts make decisions and translates the  
10 rules into terms that can be understood by a computer. Details of such rule-  
based expert systems are well understood and are disclosed in further detail in  
Principles of Artificial Intelligence, by Nils J. Nilsson (©1980 by Tioga Publishing  
Co., Palo Alto, CA) and Introduction to Artificial Neural Systems, by Jacek M.  
Zurada (©1992, by West Publishing Company, St. Paul, MN). For example, a  
15 fully-trained single-layer feedback neural network can form expert system 32,  
and a similar neural network can form each of expert systems 42.

Accordingly, Expert "Bots" 22 can be designed via expert system  
42 to develop a distinct, unique personality. Such dedicated personality enables  
the expert "Bot" 22 to make decisions based upon past experiences. The level  
20 of sophistication for an expert system can be further enhanced by adding more  
information/data to the knowledge base or by enhancing/upgrading the rules.

Scout "Bots" 20 (a-c) each comprise a set of tasks/processes that  
know how to collect necessary information in order to diagnose a system  
correctly. Additionally, Scout "Bots" 20 can be configured to process the  
25 collected results, and to make appropriate decisions based on such results. As  
such, Scout "Bots" 20 comprise self-contained software entities, or agents. For  
example, Scout "Bot" 20a can be configured to monitor the quantity of paper  
remaining in a paper tray of a printer 28 (see Fig. 1) via an associated optical  
sensor, then make decisions on whether to alert expert system 42 of a "low-  
30 paper" condition.

### **Benefits of the Solution**

The implementation of Scout "Bots" 20 within subsystems 14-18  
of diagnostic reporting system 10 provides several benefits when designing  
diagnostic reporting systems. More particularly, object oriented design

- 5 techniques and technologies can be used in generating Scout "Bots".  
Furthermore, artificial intelligence techniques can be implemented within Expert  
"Bots" 22 and diagnostic reporting system controller 30.

Object oriented design techniques enable the transformation of an  
object-oriented model into a set of specifications that are required to create a  
10 system. By expanding the model into further detail, object-oriented analysis  
moves into object-oriented design. In object oriented analysis, a problem is  
examined by modeling it as a group of interacting objects. An object is defined  
by a class, data elements and behavior. For example, in a print server  
subsystem, a print request is a class, and printing, queuing of print jobs and  
15 processing of print jobs are examples of its behavior. Objects (individual print  
requests) inherit this behavior and combine it with their own data elements. The  
use of object-oriented design techniques enables the ability to extend the  
existing diagnostic reporting systems or subsystems into new  
systems/subsystems using generalization and specialization. According to one  
20 implementation, C + + is used as the object-oriented programming language.

Artificial intelligence techniques contain the ability to learn or adapt  
through experience. Artificial intelligence includes devices and applications such  
as expert systems which are suitable for designing Expert "Bots". It is possible  
to evolve the Expert "Bots" by using theories and techniques from artificial  
25 intelligence, enabling learning from such past experiences and evolution.

Diagnostic reporting system 10 of Figure 1 is designed to be  
pluggable and extendable. Pluggable systems allow new subsystems and or  
Scout "Bots" to be added in, and refers to the ability to add a new component  
and have it work without having to perform any technical analysis or  
30 procedures. Extendibility allows the existing systems components to be  
extended using specialization/generalization methods, which enables the  
expansion or customization of capabilities. For example, extendible  
programming languages allow programmers to add new control structures, data  
types or statements.

5           Such pluggable and extendable design for diagnostic reporting systems eases maintenance difficulties, and enhances reuse of components, where each Scout "Bot" is self-contained. For example, a Multi-Function Peripheral (MFP) diagnostic system consists of a printing diagnostic subsystem (similar to subsystem 18 of Fig. 1), a facsimile diagnostic subsystem, and a  
10 scanning diagnostic subsystem. Each subsystem shares similar components, and contains similar Scout "Bots".

          In compliance with the statute, the invention has been described in language more or less specific as to structural and methodical features. It is to be understood, however, that the invention is not limited to the specific features  
15 shown and described, since the means herein disclosed comprise preferred forms of putting the invention into effect. The invention is, therefore, claimed in any of its forms or modifications within the proper scope of the appended claims appropriately interpreted in accordance with the doctrine of equivalents.

#### **Pseudocode**

20           --- About the Sample Pseudocode ---

          Sample pseudocode contains:

- \* One Expert "Bot" that monitors the print system
- \* Two Scout "Bots": paper level monitoring (2 trays - upper & lower) & paper jam monitoring

25           The sample code is setup to be as basic as possible. The usage is:

- Start up the Expert "Bot" which contains the two Scout "Bots"
- Periodically poll for the status of the Expert "Bot" & report       if there are any problems in the print system.
- Each Scout "Bot" in turn polls their own system to determine the  
30 status of the system.

          In actual implementation, the Scout "Bots" and Expert "Bots" could be set up to:

- Expert "Bot" would determine which Scout "Bots" to dynamically load at start-up time.
- 35 -- Use interrupt driven architecture to determine the status of the systems, instead of using polling. As an example, instead of periodically doing SNMP queries to determine if there is a paper jam, or the current paper level status, the print device could send out a "Trap" to indicate the change in the paper level, or to report the paper jam. In



```

5         return, the Scout "Bots" would notify the Expert "Bot" of the change in
           status.

           =====
           ===== */

/* =====
10  ===== */

    /**
        Sample Scout "Bot" to determine the paper level in the input
        trays.

        The constructor starts the paper level monitoring, and the
15     destructor stops the monitoring process. Note: the monitoring
        is done using polling.
    */

    class PaperLevel_ScoutBot
    {
20         PaperLevel_ScoutBot(char inputTrayName[]); // constructor
           ~PaperLevel_ScoutBot(); // destructor

           StartMonitoring(); // monitors the paper levels in the system

           int Status(); // reports the paper level

    private:
25         CheckPaperLevel(); // determines the paper level

           int paperLevel; // keeps track of the paper level
           char trayName[20]; // keeps track of the tray name
           int threadID; // keeps track of the thread id

    };

30     /* Constructor: perform all the necessary initializations.
           This includes starting a thread to perform the monitoring
           action for the paper levels in the input system.
    */
    PaperLevel_ScoutBot::PaperLevel_ScoutBot( char *inputTrayName )
35     {
           strcpy(trayName, inputTrayName); // keep track of the name

           /*---

```

```

5      Start a new thread to monitor the paper level in the input
      system. This allows the Scout "Bot" to live independently
      from the other Scout "Bots".
      ---*/
      threadID = StartNewThread(StartMonitoring);

10  }

/* Destructor: perform all the necessary clean ups
*/
PaperLevel_ScoutBot::~PaperLevel_ScoutBot()
{
15      /*---
      In order to stop monitoring the paper levels, it is necessary
      to kill the previously started thread. This will stop the
      "forever loop" that takes place in the StartMonitoring() code.
      ---*/
20      KillThread(threadID);

}

/* StartMonitoring: Starts the monitoring for paper levels.
*/
PaperLevel_ScoutBot::StartMonitoring()
25  {
      // poll for the paper levels forever, until it is time to exit
      while ( TRUE )
      {
30          CheckPaperLevel(); // check the current paper level
          Sleep(60*10);        // sleep for 10 mins
      }

}

/* Report: Reports the status of the paper level in the input system.
The sample code implementation reports the paper level as is;
however, this could be modified to report the paper level only
when it is low.
*/
int
PaperLevel_ScoutBot::Status()
40  {
      return paperLevel;          // this could be a % of paper level

```

```

5   }

    /* CheckPaperLevel: Gets the paper level of the input system using
       the SNMP protocol and a MIB object that corresponds to the paper
       level in the printer.
       The MIB object could be set by a sensor that is located in the
10  print device.
    */
    PaperLevel_ScoutBot::CheckPaperLevel()
    {
15        // perform the SNMP request to get the paper level
        paperLevel = GetSnmpPaperLevelFromPrinter();
    }

    /* =====
    ===== */

    /**
20    Sample Scout "Bot" to determine whether there is paper jam in the
        system.

        The constructor starts the paper jam monitoring, and the destructor stops
        the monitoring process. Note: the monitoring is done using polling.
    */

25  class PaperJam_ScoutBot
    {
        PaperJam_ScoutBot();           // constructor
        ~PaperJam_ScoutBot();          // destructor

        StartMonitoring();              // starts the monitoring for paper jams
30        BOOL Status();                // reports the paper jam

    private:
        CheckPaperJam();                // determines if there is a paper jam

        BOOL paperJam;                  // is there a paper jam?
        int  threadID;                  // keeps track of the thread id

35  };

```

```

5  /* Constructor: perform all the necessary initializations.
   This includes starting a thread to perform the monitoring action for the
   paper jams in the input system.
   */
   PaperJam_ScoutBot::PaperJam_ScoutBot()
10  {
       paperJam = FALSE;           // initialize the paper jam

       /*---
        Start a new thread to monitor the paper jam in the print
        system. This allows the Scout "Bot" to live independently
15       from the other Scout "Bots".
        ---*/
       threadID = StartNewThread(StartMonitoring);
   }

   /* Destructor: perform all the necessary clean ups
   */
20  PaperJam_ScoutBot::~PaperJam_ScoutBot()
   {
       /*---
        In order to stop monitoring the paper jam, it is necessary
25       to kill the previously started thread. This will stop the
        "forever loop" that takes place in the StartMonitoring() code.
        ---*/
       KillThread(threadID);
   }

30  /* StartMonitoring: Starts the monitoring for paper jams.
   */
   PaperJam_ScoutBot::StartMonitoring()
   {
       // poll for the paper jam condition forever, until it is time to exit
35       while ( TRUE )
       {
           CheckPaperJam(); // check if there is a paper jam
           Sleep(60*10);    // sleep for 10 mins
       }
40  }

   /* Report: Reports if there is a paper jam in the system.

```

```

5      The sample code returns if there is a paper jam in the system.
      */
      BOOL
      PaperJam_ScoutBot::Status()
      {
10          return paperJam;          // whether there is a paper jam or not
      }

      /* CheckPaperJam: Determines if there is a paper jam using
         the SNMP protocol and a MIB object that corresponds to the paper
         jam status.
15         The MIB object could be set by a sensor that is located in the
         print device.
      */
      PaperJam_ScoutBot::CheckPaperJam()
      {
20          // perform the SNMP request to determine if there is a paper jam
          paperJam = GetSnmpPaperJamStatusFromPrinter();
      }

      /* =====
      ===== */

25  /**
      Sample Expert "Bot" to monitor the printing system.
      This Expert "Bot" only contains knowledge for monitoring the paper level
      (PaperLevel_ScoutBot), and paper jam (PaperJam_ScoutBot).

      Note: construction of this object automatically starts the monitoring
30      process for the Scout "Bots" installed in the system.

      Note: This sample contains the knowledge of the ScoutBots at compile
      time (static linking), however, through the use of component
      technologies, such as COM (Component Object Model), dynamic linking
      of the Scout "Bots" at run time would be easy to do.
35  */

      class PrintSystem_ExpertBot
      {
          PrintSystem_ExpertBot();          // constructor
          ~PrintSystem_ExpertBot();         // destructor

```

```

5      Status();                      // reports the print system status

    private:
        PaperLevel_ScoutBot    *paperLevelBot[2]; // assumes we have 2
                                                // paper trays in the
                                                // printing system
10      PaperJam_ScoutBot      paperJamBot;      // monitors the paper
                                                // jams
    };

    /* Constructor: perform all the necessary initializations.
15      This construction will automatically cause the Scout "Bots" to monitor
      their systems.
      Note: PaperJam_ScoutBot is automatically constructed.
    */
    PrintSystem_ExpertBot::PrintSystem_ExpertBot()
20    {
        // construct the paper tray Scout "Bot" for "upper" tray
        paperLevelBot[0] = new PaperLevel_ScoutBot("upper tray");
        // construct the paper tray Scout "Bot" for "lower" tray
        paperLevelBot[1] = new PaperLevel_ScoutBot("lower tray");
25    }

    /* Destructor: perform all the necessary clean ups
      Note: This will automatically cause the Scout "Bots" to stop monitoring
      their systems.
    */
30    PrintSystem_ExpertBot::~PrintSystem_ExpertBot()
    {
        // clean up all the Scout "Bots"
        delete paperLevelBot[0];
        delete paperLevelBot[1];
35    }

    /* Status: Reports the status of the print system.
      Analyzes the Scout "Bots" status information, and returns the most
      critical error to the user. The return value is a string that can be displayed
      to the user.
40    In this sample:

```

- 5       - Paper jam is higher priority then paper out.
- Paper levels > 25% doesn't get reported to the user
- Upper tray is assumed to be default tray, and its low paper condition is reported to the user before lower tray.

```

*/
10 char *
PrintSystem_ExpertBot::Status()
{
    // get the status of the paper level
    int upperTrayLevel = paperLevelBot[0]->Status();
15    int lowerTrayLevel = paperLevelBot[1]->Status();

    // get the status of paper jam
    BOOL isPaperJam = paperJamBot.Status();

    // determine if there is a problem in the system
    if ( isPaperJam == TRUE )
20    {
        return ( "There is a paper jam!" );
    }
    else if ( upperTrayLevel == 0 )
    {
25        return ( "Out of paper in upper tray!" );
    }
    else if ( lowerTrayLevel == 0 )
    {
30        return ( "Out of paper in lower tray!" );
    }
    else if ( upperTrayLevel < 25 )
    {
        return ( "Low paper condition in upper tray!" );
    }
35    else if ( lowerTrayLevel < 25 )
    {
        return ( "Low paper condition in lower tray!" );
    }

    return NULL; // no error conditions found
40 }

/* =====
===== */

/**
Main code to start & stop the print system monitoring

```

```

5  ***/

void
main( int ac, char *av[] )
{
    /*---
10      1st construct the Expert "Bot" to monitor the printing system.
      Note: the construction of the PrintSystem_ExpertBot automatically
      starts the monitoring process (which uses polling).

      In this sample, this is the top tier Expert "Bot"; however, it
      could be designed as a 2nd tier Expert "Bot" when combined with
15      other Expert "Bots", such as workstation monitoring system.
    ---*/
    PrintSystem_ExpertBot    printSystemExpert; // construct the object

    char *statusString;

    while ( user_selected != ExitPrintSystemExpert )
20    {
        /*---
          Get the status of the print system & display it to the
          user. printSystemExpert.Status() returns NULL if there
          is no problems to report in the print system.

25          DisplayToUser() function is basically a printf, or
          some sort of MessageBox to the user.
        ---*/

        if ( ( statusString = printSystemExpert.Status() ) != NULL )
        {
30            DisplayToUser( statusString );
        }
        Sleep(60*10);           // sleep for 10 mins
    }

    /*---
35      Exiting the program will automatically perform the necessary
      clean up functions for the Scout "Bots", and it will stop
      the monitoring process.
    ---*/
}

```



CLAIMS

What is claimed is:

- 1                   1.       An apparatus (10) for independently and/or simultaneously  
2       monitoring functionality of peripheral components within a computer network  
3       environment (12), comprising:  
4                   a computer (25);  
5                   a peripheral component (24, 26, 28) signal coupled with the  
6       computer (25); and  
7                   a plurality of agents (20), at least one agent (20) associated with  
8       the peripheral component (24, 26, 28) and operative to collect information  
9       needed to diagnose functionality of the peripheral component (24, 26, 28).
- 1                   2.       The apparatus (10) of claim 1 wherein the agents (20) each  
2       comprise a self-contained software entity.
- 1                   3.       The apparatus (10) of claim 1 wherein the agent (20) is  
2       implemented as an object oriented program.
- 1                   4.       The apparatus (10) of claim 1 wherein the agent (20) is  
2       implemented as an object oriented program.
- 1                   5.       The apparatus (10) of claim 4 wherein each peripheral  
2       component (24, 26, 28) comprises a controller (40) configured to implement a  
3       subsystem expert system (42), the expert system (42) configured to monitor  
4       and analyze results directly from the agents (20) so as to provide a diagnostic  
5       subsystem that collects, analyzes and/or reports results from the agents (20) to  
6       the central expert system (32).
- 1                   6.       The apparatus (10) of claim 1 wherein each peripheral  
2       component (24, 26, 28) comprises a controller (40) configured to implement a

3 subsystem expert system (42), the expert system (42) configured to monitor  
4 and analyze results directly from the agents (20) so as to provide a diagnostic  
5 subsystem that collects, analyzes and/or reports results from the agents (20) to  
6 the computer (25).

1           7.     The apparatus (10) of claim 1 wherein the computer (25)  
2 comprises a diagnostic reporting system controller (30) configured to implement  
3 a central expert system (32), and the peripheral component (24, 26, 28)  
4 comprises a subsystem controller (40) configured to implement a subsystem  
5 expert system (42).

1           8.     A method for diagnosing and reporting peripheral  
2 functionality for peripheral components (24, 26, 28) contained within a network  
3 environment (12), comprising the steps of:  
4           providing at least one autonomous agent (20) associated with a  
5 peripheral component (24, 26, 28) and operative to monitor peripheral  
6 component functionality and capable of reporting such monitored functionality  
7 to a user of the network environment (12);  
8           concurrently monitoring functionality of each peripheral component  
9 (24, 26, 28) with at least one associated agent (20);  
10          diagnosing peripheral component functionality by evaluating the  
11 monitored peripheral component functionality; and  
12          alerting a user to the diagnosed peripheral component functionality  
13 when a condition warranting the alerting of the user is determined, for each  
14 peripheral component (24, 26, 28) contained within the network environment  
15 (12).

1           9.     The method in accordance with claim 8 wherein the  
2 autonomous agent (20) comprises an autonomous program operative to monitor  
3 an operating characteristic of a peripheral component (24, 26, 28).

- 1                    10. The method in accordance with claim 8 wherein the step of
- 2 diagnosing peripheral component functionality comprises the step of applying a
- 3 knowledge base from an expert system to process results from the autonomous
- 4 agents (20).



Application No: GB 9906578.1  
Claims searched: 1-10

Examiner: Melanie Gee  
Date of search: 21 October 1999

**Patents Act 1977**  
**Search Report under Section 17**

**Databases searched:**

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:  
UK CI (Ed. Q): G4A (AFMD, AFMX)  
Int CI (Ed. 6): G06F 11/30, 11/32, 11/34; H04L 12/26  
Other: Online: WPI, EPODOC, COMPUTER

**Documents considered to be relevant:**

Category	Identity of document and relevant passage	Relevant to claims
X	EP 0822498 A1 (BULL), see especially abstract and claim 1.	1-4, 8 & 9
X, P	US 5872931 A (CHIVALURI), see whole document.	1-4, 8 & 9
X	US 5655081 A (BONNELL et al.), see whole document.	1-6 & 8-10.

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.